On-Board Model Based Fault Diagnosis for CubeSat Attitude Control Subsystem: Flight Data Results

Ryan Mackey, Allen Nikora,
Cornelia Altenbuchner, Robert Bocchino,
Michael Sievers, Lorraine Fesq

Jet Propulsion Laboratory
California Institute of Technology

Ksenia O. Kolcio, Matthew J. Litke,
Maurice Prather

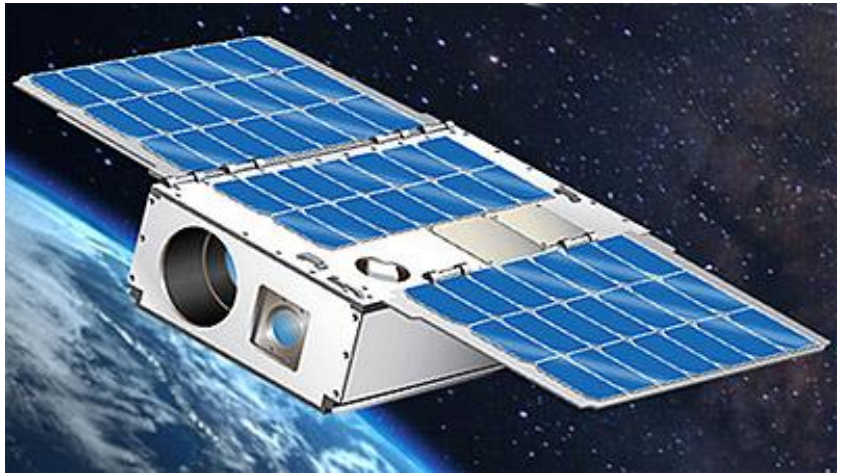
okean solutions

Government sponsorship acknowledged.

Project Overview

The ASTERIA Spacecraft provided a unique opportunity to demonstrate on-board model-based diagnosis supporting complete on-board autonomy

- Extended Mission Experiment: Ability and willingness to host reasoning software as payload
- Capable spacecraft with complex subsystems and behavior (ACS chosen for study), able to gather full rate data
- Mature testbeds and test procedures
- Modular flight software architecture (F Prime)



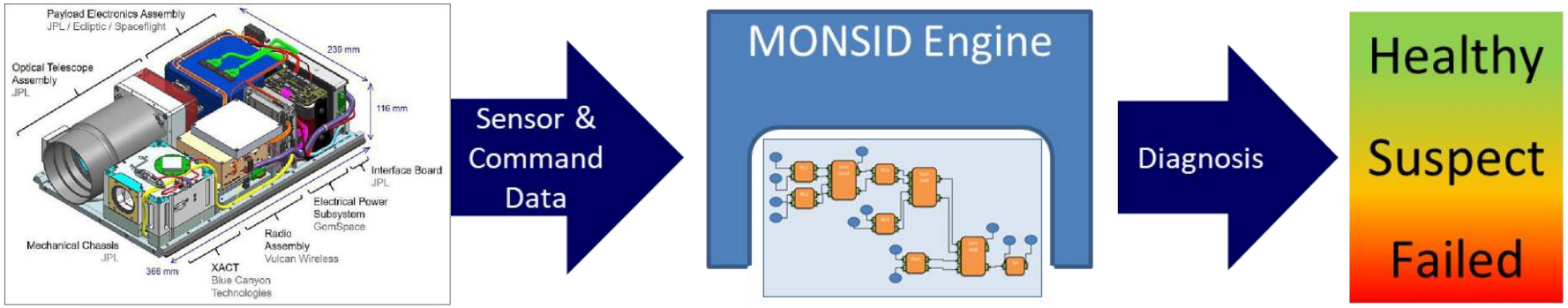
Objectives

- Demonstrate MONSID reasoner on ASTERIA XACT Attitude Control Subsystem (Blue Canyon Technologies)
- Model XACT and integrate with ASTERIA flight software
- Perform robustness and flight readiness tests
- Benchmark computing resource impacts
- Demonstrate on ASTERIA System Testbed (with captured data) and on board the spacecraft
- Include MONSID in demonstration of closed-loop autonomy

Goals

- Provide a *trusted, reusable, model-based* on-board diagnostic reasoning capability
- Detect failures and mission threats
- Estimate remaining capability in real time
- Enable on-board autonomy to respond to unknown events

MONSID Background



MONSID (Model-based **O**ff-**N**ominal **S**tate **I**dentification and **D**etection**)** consists of:

- Diagnostic Engine (application independent)
- Model capturing nominal system behavior (application specific, network of hardware components)

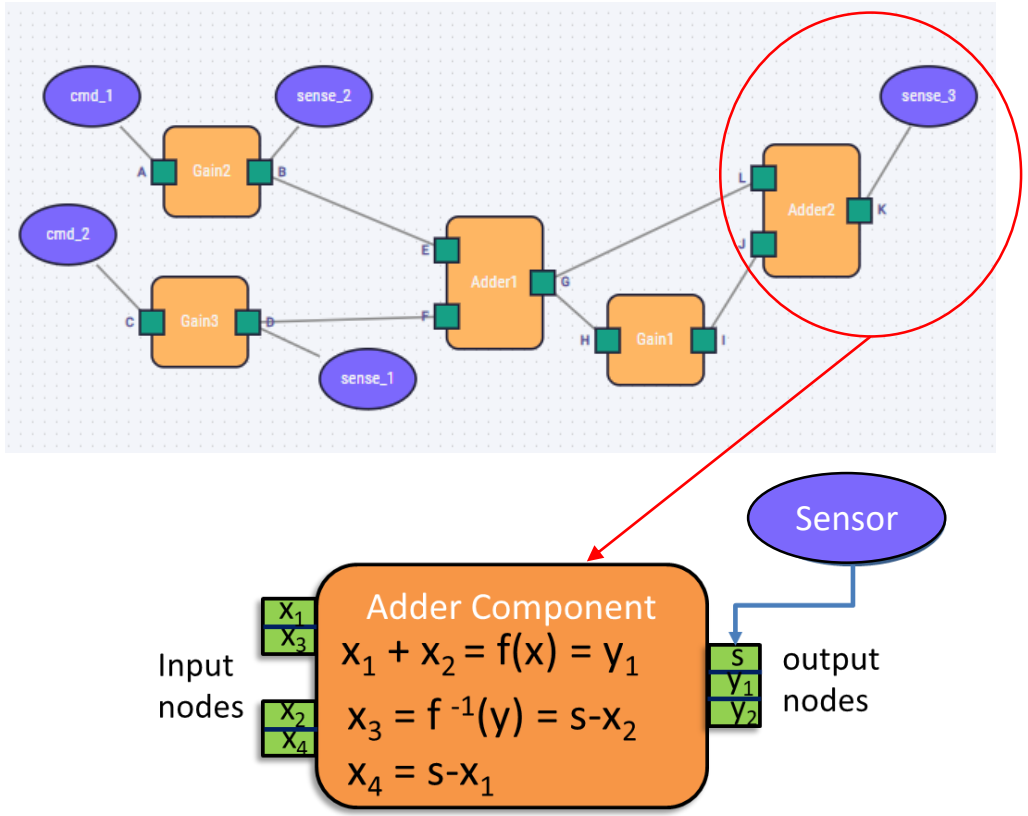
- Sensor and Command data are inputs to MONSID Model
  - Inputs are propagated through model, producing multiple estimates of state variables
  - Diagnostic engine flags inconsistencies in state variable estimates
  - Utilizes Constraint Suspension technique to detect and isolate faults

- MONSID provides a synchronous report of healthy/failed system state
- Compact, modular API capable of operating on resource-limited platforms

MONSID Model Description

- Components:
  - Associated with system state variables (vertices) and *constraints* that capture behavior
  - Components are not entirely stateless, but may include some "memory" of previous state estimates
  - Components have a notion of "input" (left) vs. "output" (right), roughly equivalent to functional flow
- Constraints:
  - Constraints are equations that relate state variables to one another
  - Forward constraints capture input to output behavior
  - *Reverse constraints* (output to input) are also needed to verify model consistency
  - No restrictions on constraint format
- "Sensors:"
  - Provide state estimates to model from physical system (e.g., sensor values)
  - Commanded and reported hardware data, FSW parameters

Model Layout



- Input and output nodes:
  - Each holds two or more versions of a state variable estimate for consistency checks
  - Faults are indicated when discrepancies exceed a tolerance threshold

XACT Modeling Approach

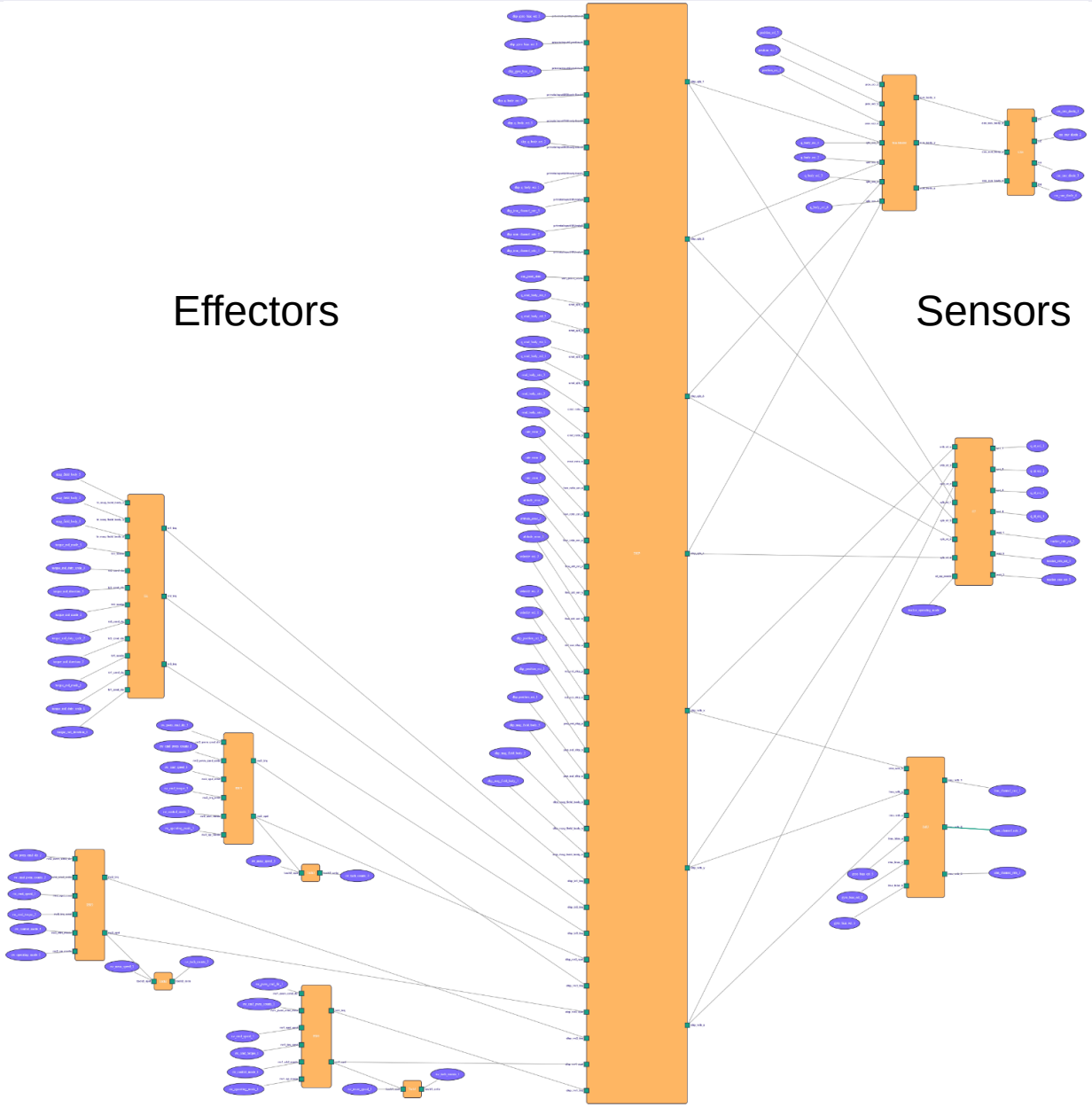
- Starting assumptions: Bounding the system under study (in this case, restricted to failures of XACT components)
  - ❖ Identify relevant telemetry and control signals
  - ❖ Break up model into components consistent with our desired level of diagnosability and data availability
  - ❖ Associate signals with components – follow system state effects model, and introduce hidden state variables (viz., internally computed only) as needed
  - ❖ Where necessary create “pseudocomponents” – components that model dynamic and environmental state, needed by other components. These do not represent physical hardware.
  - ❖ Model complexity trades: 1. Minimizing connections leads to leaner models; 2. Sensor placement determines diagnosability.

- Develop constraints from first principles and example data, one component at a time
  - ❖ Develop both forward constraints (predictive) and reverse constraints (deductive)
  - ❖ Constraints need not be symbolic inverses of each other – apply workarounds if reverse constraints cannot be calculated or do not exist
  - ❖ Need to account for different spacecraft modes and hardware operational modes
  - ❖ Verify that constraints are protected against bad data or numerical singularity / divide-by-zero conditions
  - ❖ Check constraints against test data, one component at a time
  - ❖ Begin with simple constraints, and add complexity as needed to increase fidelity

XACT Model Summary

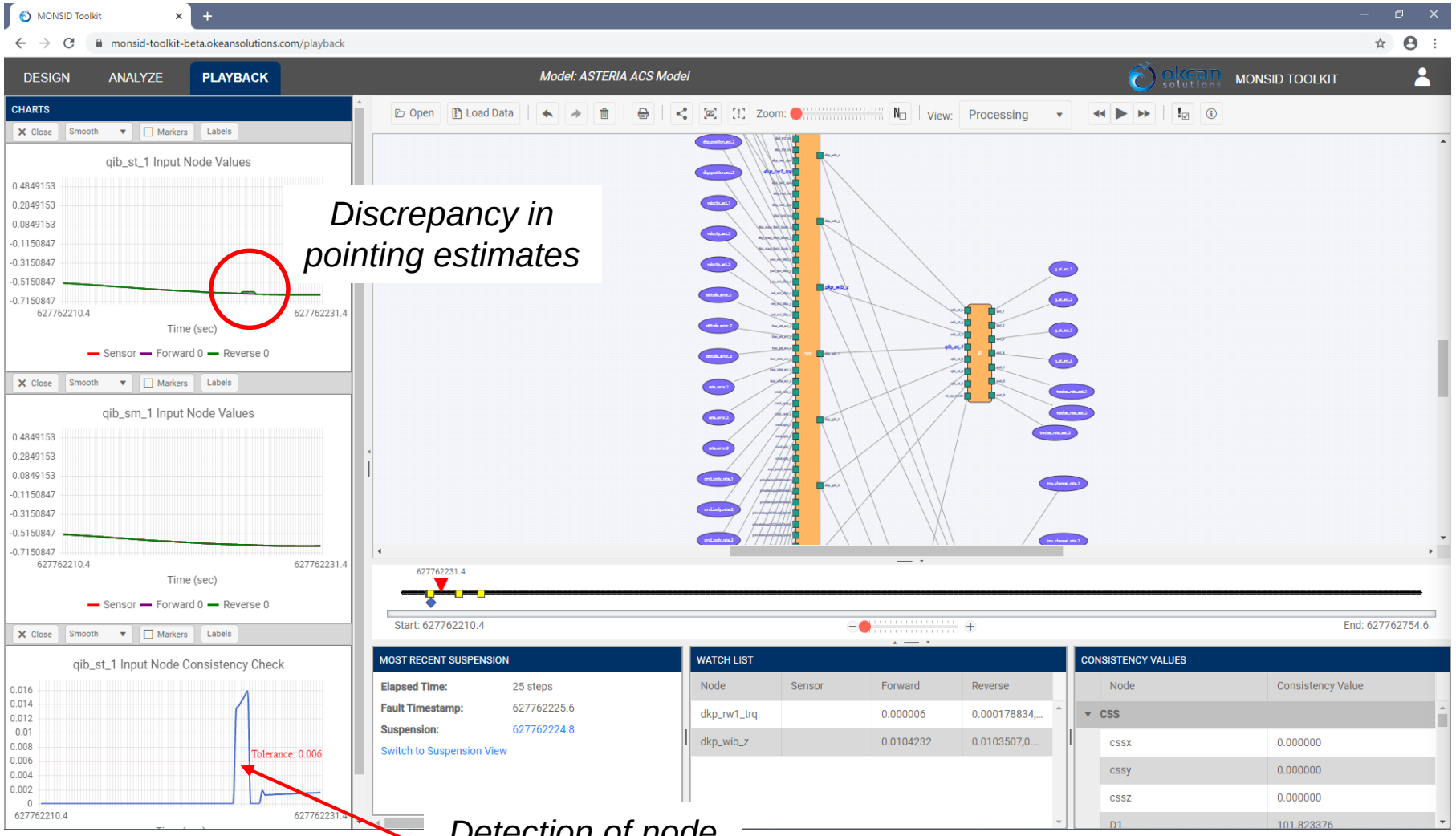
- Prove that modeling a real subsystem (BCT XACT) is feasible, affordable, and effective
- Individual, reusable model components created and unit-tested first
- Components placed into system topology, as shown here

- Final model size:
  - ❖ 99 input nodes (94 sensor / command inputs plus 5 hidden state variables)
  - ❖ 36 output nodes
  - ❖ 12 components:
    - 3x Reaction Wheels
    - 3x Tachometers
    - 1 Magnetorquer block
    - 1 Sun Sensor
    - 1 IMU
    - 1 Star Tracker
    - 1 Sun Visibility Model (pseudocomponent)
    - 1 Dynamics and Kinematics pseudocomponent



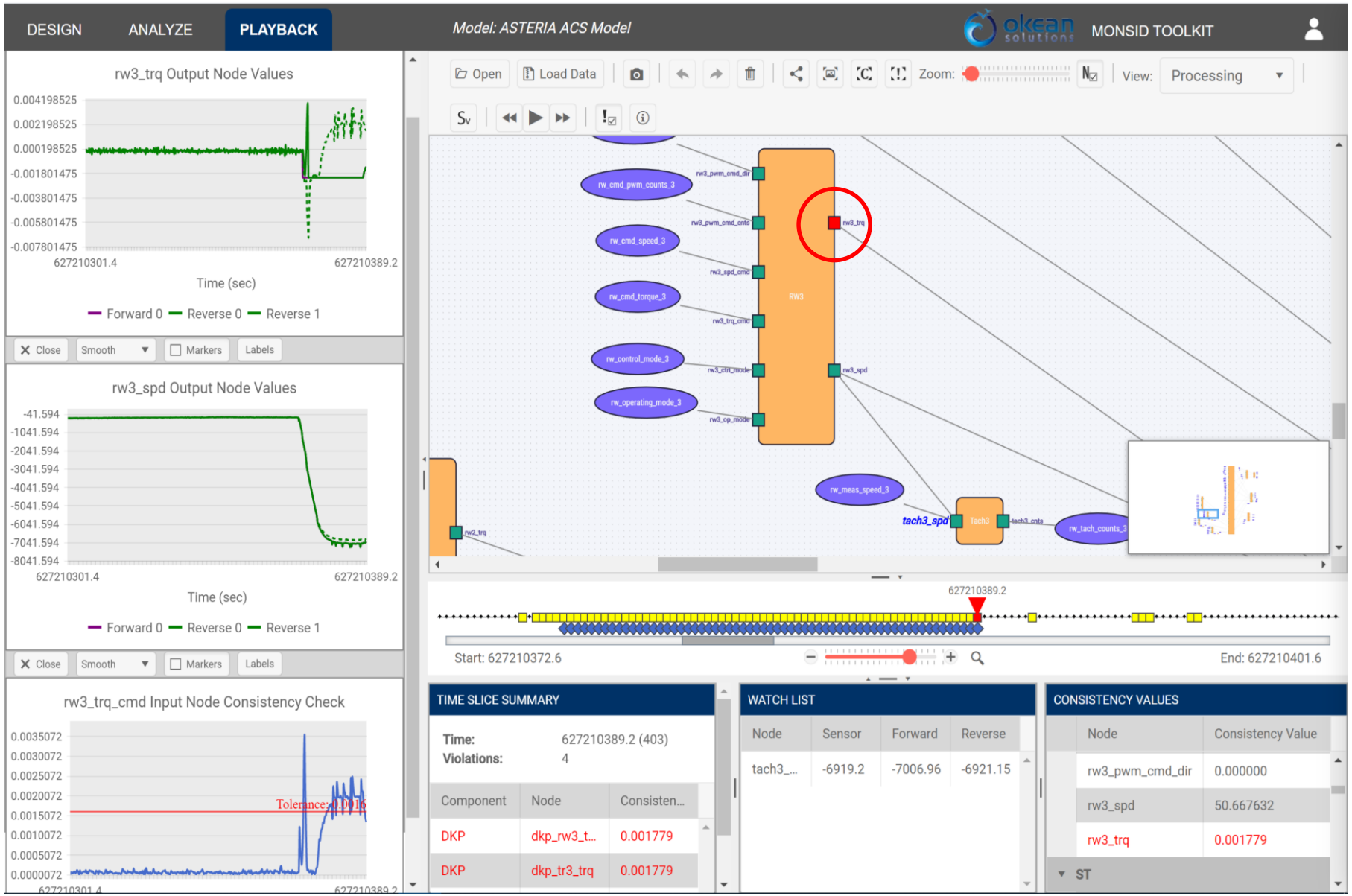
Model Testing: Flight Data

- After completion, tested model with captured data from ASTERIA – revealing previously unknown events
- Results verified by running MONSID in developer mode and as part of ASTERIA FSW on ASTERIA System Testbed
- **Shown here**: Detection and isolation of a (very) brief pointing knowledge inconsistency during slew
- Caused by momentary change in Star Tracker state providing an incorrect transient output (ignored by XACT algorithms, but still reported)
- Similar behavior has been seen on other missions



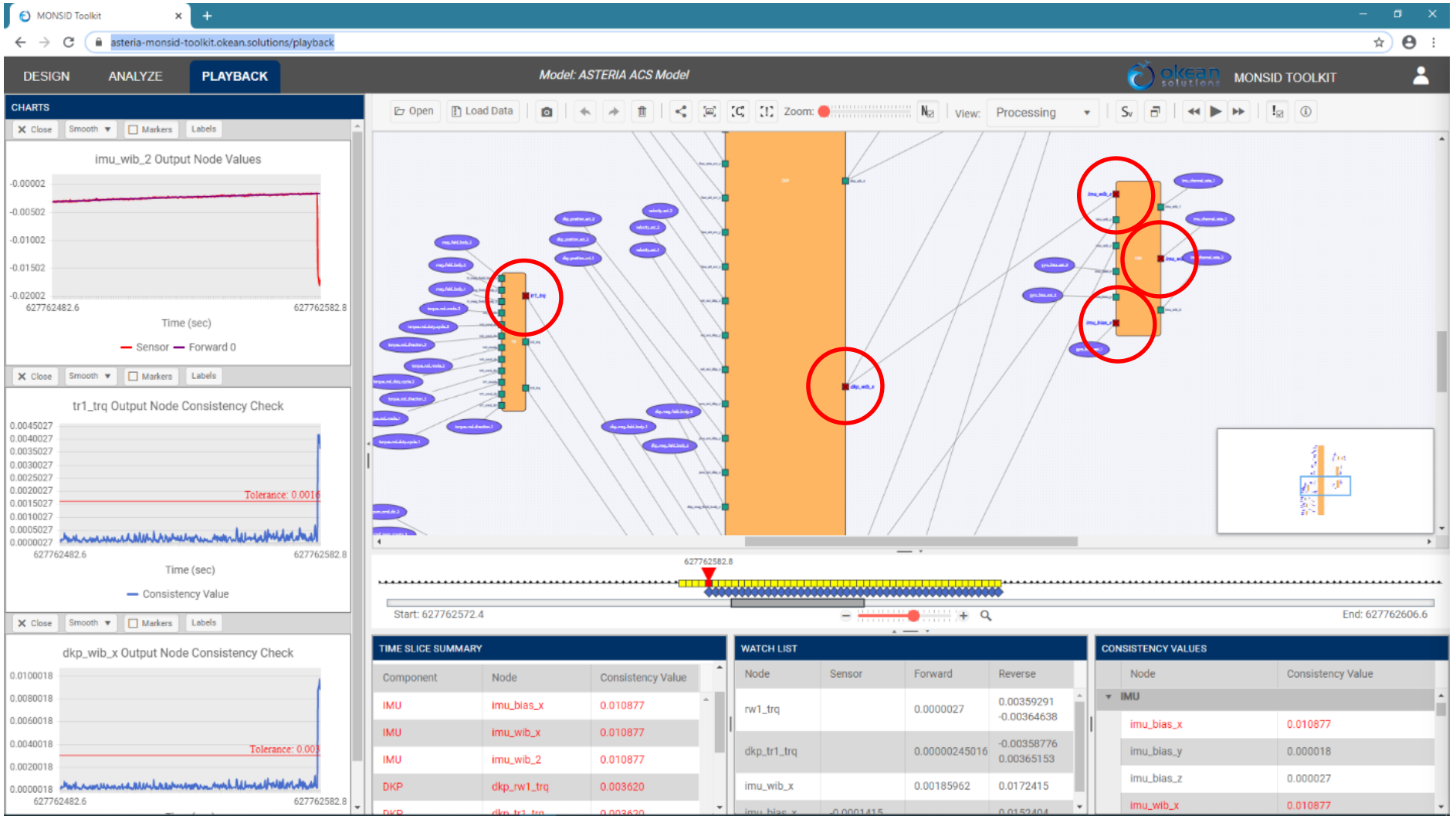
Model Testing: Flight Data

- **Shown here**: Detection and isolation of suspected reaction wheel saturation discovered in captured flight data
- Detected discrepancies include mismatch in actual torque, as estimated from torque command and wheel RPM
- Isolation step confirms that only one wheel (RW3) is involved, and no other hypothesis explains this observation



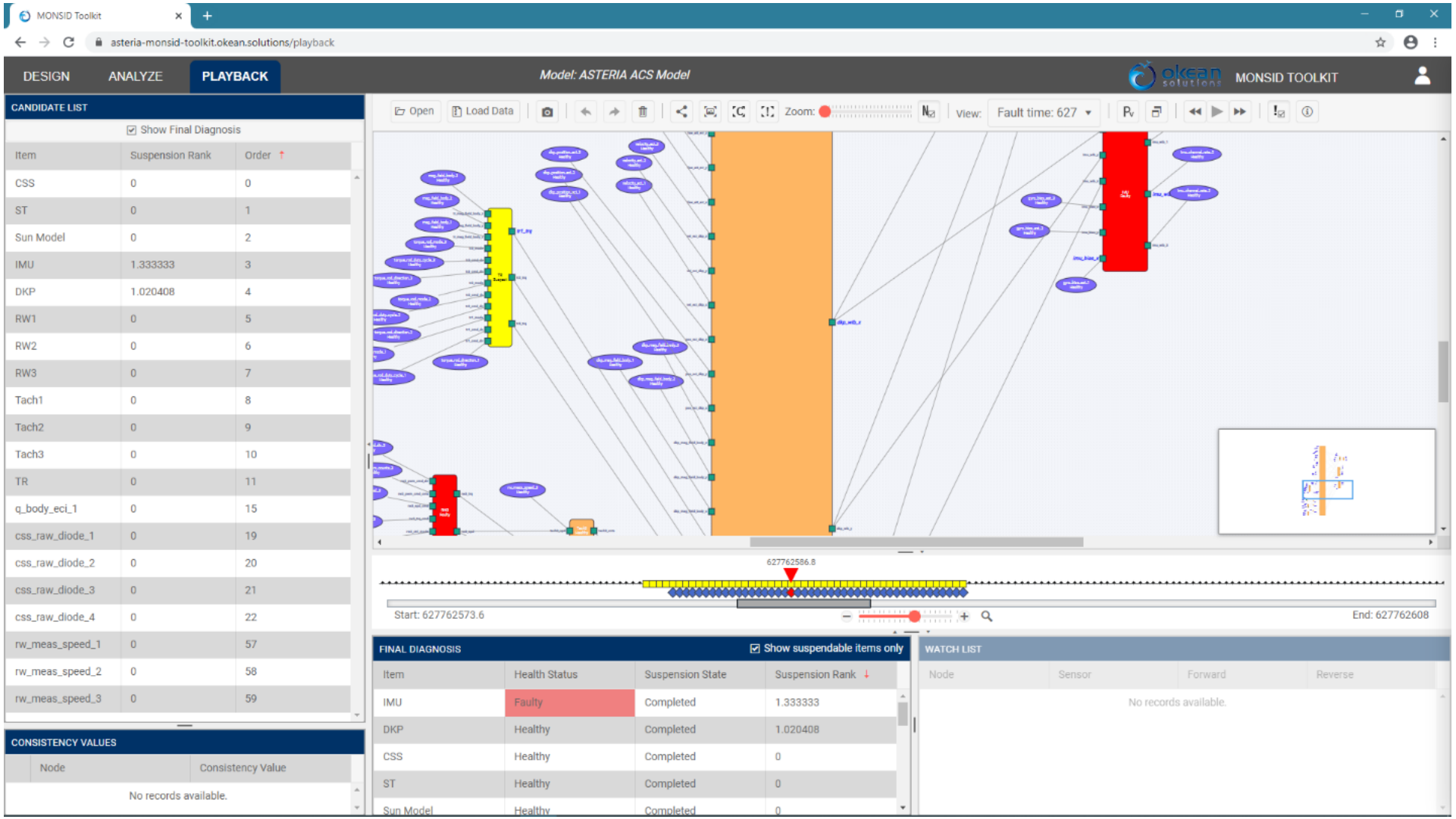
Model Testing: Seeded Faults

- To enable safe fault detection testing on board, MONSID FSW was provisioned with a commandable fault injection capability
- This biases sensor data prior to analysis by MONSID
- **Shown here**: Bias introduced in one IMU channel
- Because such a change *should* propagate throughout the system, this results in several discrepancies, as the model now predicts a different overall state
- This poses an interesting challenge for isolation



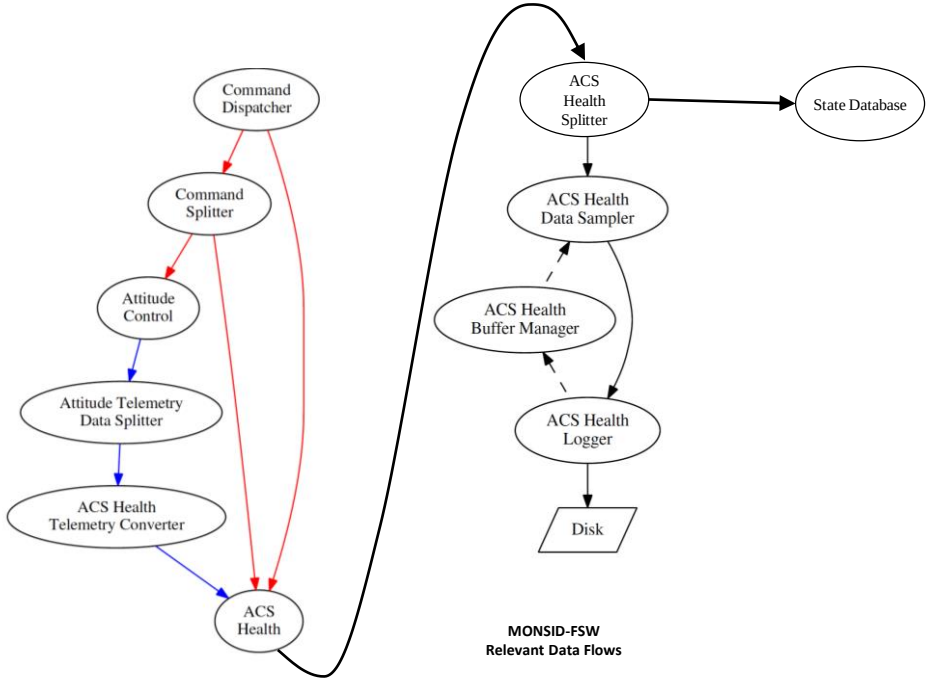
Model Testing: Seeded Faults

- In this case, the IMU is flagged as the most probable source of the fault
- However, other components could, in theory, lead to this outcome
- MONSID also found the degenerate case where RWAs had all failed in a way that cancelled out other potential signals, or in a way that a torque rod problem could mask
- This is of low probability, but technically correct – will be filtered out in updates to the MONSID engine

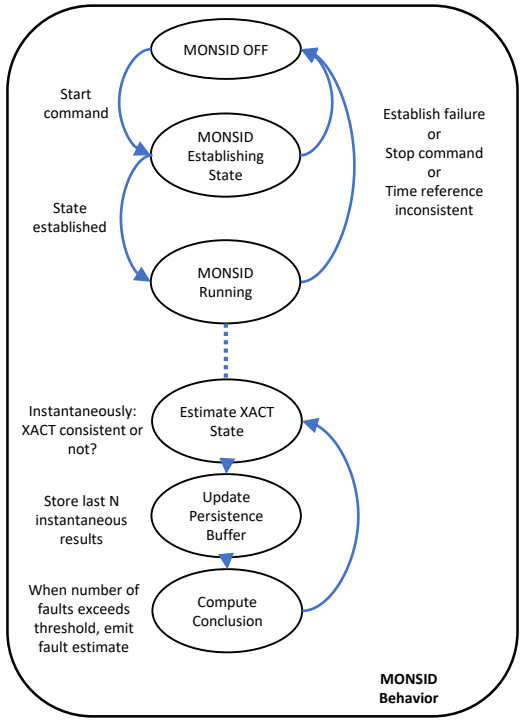


Flight Software Integration

- ASTERIA's F Prime-based flight software enabled an effective method of MONSID execution inside the control loop
  - ❖ MONSID primarily consumes data from the Attitude Control Components (the XACT ACS manager), but may also access commands sent to ACS
  - ❖ "Splitter components" used to send data from sources to MONSID without interrupting existing dataflows
  - ❖ MONSID implemented in a fully flight-ready configuration, sending summary results through telemetry while saving detailed information to disk for verification purposes
- MONSID itself managed by a new F Prime component (the ACS Health component)
  - ❖ Implemented a very simple behavior for MONSID, requiring only two commands (*Start* and *Stop*)
  - ❖ This approach can be retrofitted to many F Prime FSW deployments



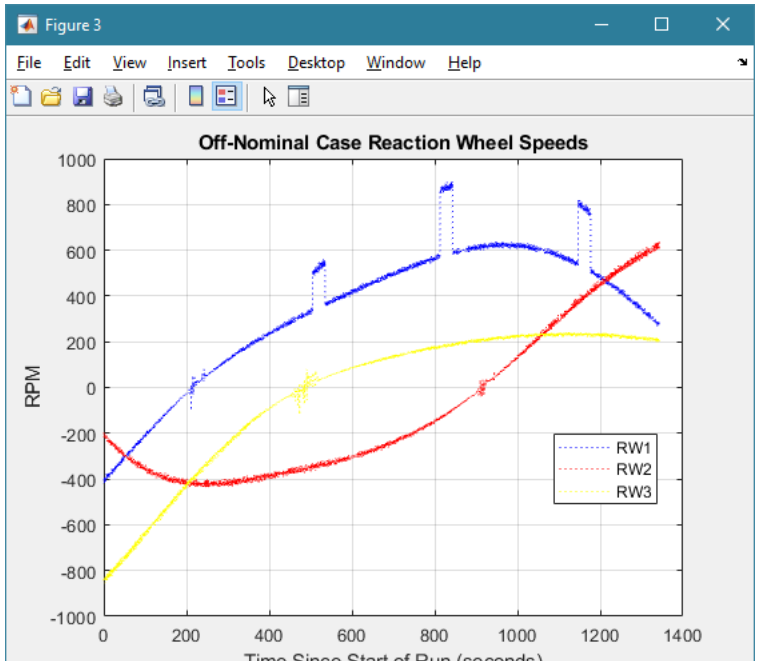
MONSID-FSW Relevant Data Flows



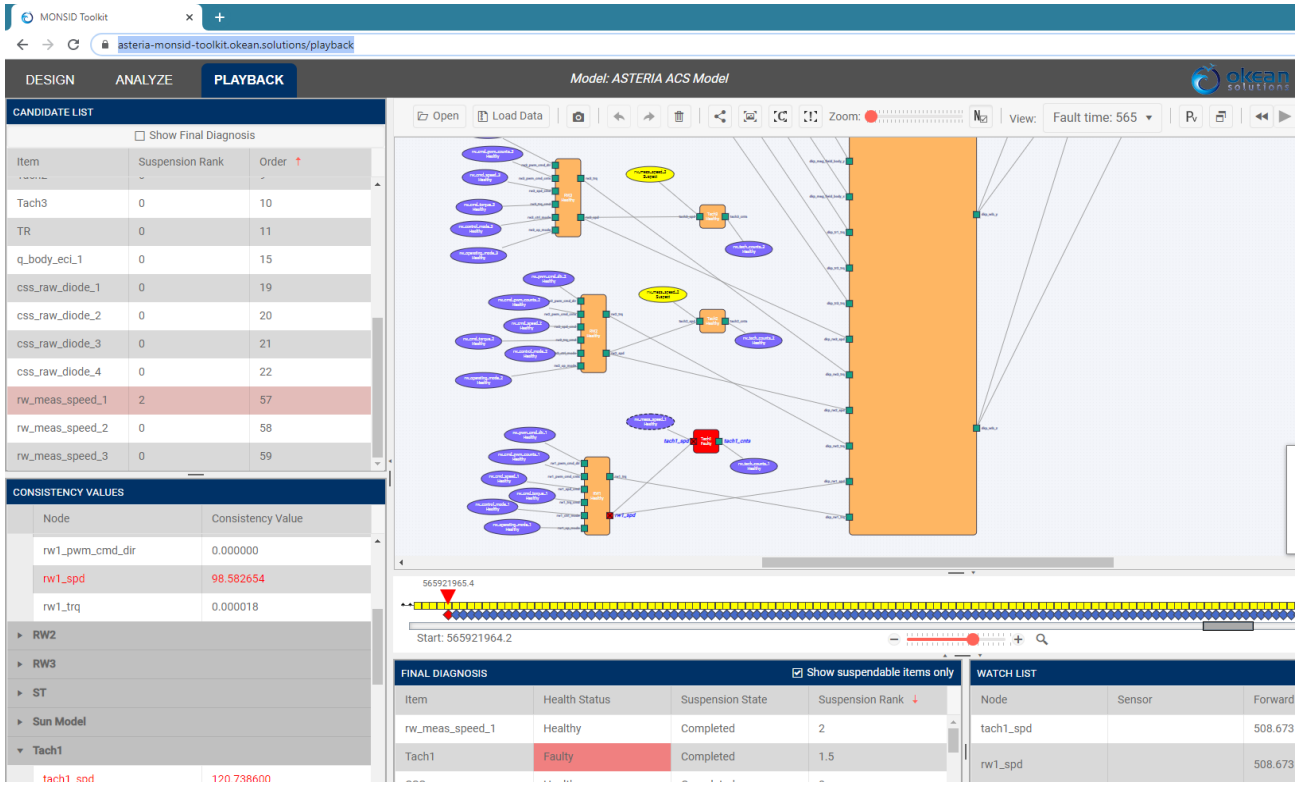
MONSID Behavior

Closed-Loop Autonomy

- MONSID-FSW was also combined with two other reasoners (AutoNav and MEXEC) to demonstrate closed-loop ability to ensure activity completion
  - ❖Combined demonstration would exceed computational limits of ASTERIA avionics
  - ❖Tested instead on workstation testbed, leveraging MONSID fault injection capability



The three notches are injected faults of the RW1 tachometer
Noise events on RW1 and RW3 are real…



MONSID Toolkit displaying detection and diagnosis result during first fault episode
Red box indicates the diagnosis (RW1 Tachometer)

Other Findings

- *Spacecraft Behavior:*
  - ❖ (Re-)Discovered that spacecraft magnetic torque due to its quiescent dipole was much higher than expected
  - ❖ Used the onboard magnetometer estimate instead of model predicts in the MONSID model, departing from the XACT control laws
- *Testbed Idiosyncrasies:*
  - ❖ In verification testing, discovered a (harmless) difference in NaN handling between software and avionics testbeds
- *Performance Benchmarking:*
  - ❖ Verified that MONSID impact on system resources is modest – 160 kB code base, 1.2 MB increase in resident memory, and < 5% CPU in worst case testing

Conclusions

- While a mission-ending spacecraft anomaly prevented actual on-board testing, ASTERIA still provided an excellent maturation opportunity for model-based fault management
  - ❖ Enabled by access to mission resources, from controllers and subsystem experts to software and hardware testbeds, from established mission processes and norms to flight data on demand
- MONSID demonstrated accuracy and reliability on captured flight data
  - ❖ Focus on testing and test realism resulted in numerous discoveries, including spacecraft behaviors and unknown benign faults, latent defects in system testbeds, hidden vulnerabilities in technology
- Demonstrated a functioning flight software implementation, feasible within the requirements and resources of a typical flight project
  - ❖ Modular flight software architecture, provided by F Prime, proved sufficiently flexible to support model-based fault diagnosis, and simplified integration and test
  - ❖ This approach can be applied to many other missions using F Prime
- There are no remaining hurdles to implementing MBFM in flight